I.E.S BARAJAS

PlayMesa

Proyecto basado en Spring Framework

Pedro José Rivera Pérez

11/06/2020

Índice

1. DESCRIPCIÓN GENERAL

El proyecto **PlayMesa** nace para dar respuesta a una necesidad, la categorización de juegos de mesa desde un punto de vista más **pedagógico**. Así pues, paso a hacer una breve reflexión del marco educativo en el que, desde mi punto de vista, nos encontramos.

La idea de "escuela" que conservamos desde finales del siglo XIX, enfocada a las necesidades de la industrialización, ha dado de sí todo lo que podía dar. Los objetivos con que nació y se estructuró están cada día más lejos de la realidad actual. La educación consiste en conseguir que el alumnado aprenda y cada día sabemos mucho más sobre cómo aprende nuestro cerebro; unas evidencias que todavía cuestionan más el actual modelo educativo.

El juego tiene indiscutibles beneficios, permitiendo que en cada uno de nosotros crezca equilibradamente el cuerpo, la inteligencia, la afectividad y la socialización. Es fuente de placer, alegría, curiosidad, descubrimientos, retos, aprendizaje y satisfacción. Los beneficios de aprender jugando en familia son múltiples. Además de estimular habilidades como la creatividad, la imaginación, la capacidad de socialización o el razonamiento lógico, jugar en familia refuerza los vínculos emocionales entre padres e hijos. Por último, no olvidemos también la importancia del juego como transmisor de valores. A través de él, podemos enseñar desde un primer momento la importancia de valores como el respeto, la tolerancia, la generosidad o la confianza en los demás.

El juego es curiosidad y nos permite descubrir, experimentar, explorar y manipular el entorno; obteniendo como beneficio colateral al juego el aprendizaje. Así, el juego se convierte en la forma natural de aprender, pues aprendemos a través de los sentidos, las emociones y el movimiento.

Tuve el privilegio de asistir en mi educación primaria a un colegio María Montessori. Recuerdo entrar con un pequeño problema de dislexia, tenía dificultades para diferenciar la 'd' y la 'b', algo que detectaron rápidamente y consiguieron corregir gracias a sesiones especiales en las que, mediante juegos, aprendía el uso de dichas letras. No fueron necesarias muchas sesiones, en seguida pude adaptarme al ritmo de la clase, algo que era fundamental, ya que tu conocimiento avanzaba en conjunto con el de tus compañeros, tratándose de grupos en los que además estábamos mezclados tres cursos diferentes.

Recuerdo que jugando con mapas políticos en divertidas competiciones, al final de aquel año me sabía todos los países y capitales del mundo divididos por continentes, así como, jugando con el ábaco conseguía dividir y multiplicar cifras impensables en aquel momento para mí.

Disponíamos de enormes tablas con todas las clasificaciones del reino animal y vegetal donde tenías que colocar fichas con distintas especies y variedades, y digo disponíamos ya que eras tú mismo el encargado de organizar tu tiempo, tus objetivos y tu material (lógicamente siempre había supervisión, pero tenías cierta sensación de control y responsabilidad al decidir cuándo querías estudiar matemáticas y cuando ciencias naturales, por poner un ejemplo), había una carga de madurez y conocimientos que en la educación tradicional o se ven mucho más adelante o no se llegan a ver.

Así pues, viendo la necesidad que tiene la educación por buscar recursos alternativos, pienso que los juegos de mesa siguen siendo junto a la tecnología una de las apuestas más interesantes actualmente a disposición de los docentes. En contra posición a esto no hay referentes en la red que proporcionen información de forma objetiva acerca de dicho material, la única alternativa es la propia información que otorgan las editoriales de dichos juegos, consultar en foros o algún artículo de presa especializada suelto.

En este punto, uno se podría preguntar, ¿es que no existen bases de datos de juegos de mesa? La respuesta es sí, pero la pregunta que habría que hacerse es, ¿puedo obtener de dichas bases de datos juegos, por ejemplo, que mejoren la memoria? En este caso la respuesta sería no. Y es ahí donde **PlayMesa** tiene algo que aportar.

Hemos visto un poco la situación en la formación y educación, ahora paso a hacer un análisis de la situación actual del mercado de los juegos de mesa.

Estamos en la época dorada de los juegos de mesa, esta afirmación puede sonar muy rotunda, pero, por ejemplo, datos como el de una de las principales empresas del sector "*Devir*" nos demuestran que sus ventas han aumentado en un 40% año tras año, y llevan así desde 2014.

Cada vez existen más ferias especializadas y otras que adoptan a los juegos de mesa entre su oferta de contenido. En España tenemos hasta ocho grandes ferias dedicadas a ellos además de los cientos de eventos que se producen cada fin de semana en las principales capitales de nuestro país.

Sólo en España los juegos de mesa y los puzles facturan 90 millones de euros, cifra que se eleva hasta los 2.200 millones en el conjunto de Europa. Estos son datos de 2017 que se han obtenido en "The NPD Group". Este mismo grupo, nos ofrece información actual, de cómo el 45% de los españoles afirma que desde que comenzó el confinamiento dedica más tiempo a jugar que antes con juegos y juguetes (excluyendo videojuegos) un comportamiento más marcado en España que en otros países europeos, como Francia (37%), el Reino Unido (31%) o Alemania (27%).

Con estas premisas, queriendo acercarme al mundo *Spring* y más concretamente a sus módulos de *Thymeleaf, Web, Spring Security, Spring Data Jpa, Spring Mail* y el despliegue en *Cloud,* me inicié en este proyecto que se puede considerar ya de cierta envergadura.

Quiero dejar constancia de que, aunque la aplicación actualmente tiene como principal objetivo la clasificación y análisis de juegos de mesa, esta podría pivotar en cualquier momento y con relativa facilidad a otros objetivos y temáticas como libros o productos multimedia. Precisamente esta es una de las bondades que nos ofrece trabajar con un "*framework*", los principales componentes de la aplicación son totalmente reusables y únicamente cambiaría nuestro modelo de datos y vistas asociadas.

El uso de *Spring* en vez de *JSF* por ejemplo, no es casual, solo hay que hacer una rápida búsqueda por la web de tutoriales de cualquiera de los dos. *Spring* supera con creces en número de ejemplos, la claridad de los mismos y sobre todo lo actualizados que están. Además inicializadores como *Spring Boot* hacen muy cómodo su uso.

2. ANÁLISIS Y DISEÑO BBDD

La forma tradicional de construcción de software empieza con un diseño conceptual de la base de datos, de hecho, esta ha sido la manera en que he hecho mis proyectos hasta ahora, a grandes rasgos construía mi modelo de datos y posteriormente mi diagrama de flujo. Poco a poco intentaba ir afinando más en detalles hasta conseguir un modelado más extendido que me diese la seguridad necesaria para empezar a escribir código.

Con este proyecto he llevado otro proceso de diseño, gracias a la experiencia previa que me ha dado el uso de JPA en el entorno lectivo. He dejado que sea el asistente de *Spring Data* quien modele la base de datos en función de las anotaciones utilizadas en el código.

Dichas anotaciones no solo crearán los campos necesarios en la base de datos, sino también las reglas que rigen dichos campos, por ejemplo, podemos dejar que el sistema autogenere las claves primarias o determinar qué tipo de asociación e incluso determinar cuál es nuestra entidad "fuerte" y "débil" dentro de dicha asociación.

Empecé el diseño buscando qué entidades podían intervenir en el modelo, en un inicio tenía seis entidades claras que pertenecerían al modelo: Juego, Editorial, Categoría, Habilidad, Usuario y Rol.

Estos fueron mis pilares para la construcción de un prototipo, el cual tardó poco en empezar a ampliarse y modificarse. Este punto es interesante, ya que en ella radica una de las principales diferencias y ventajas con respecto de la forma en la que diseñaba antes.

La flexibilidad de que el modelo de datos dependa totalmente de la aplicación me permitía ir pensando en funcionalidades de la misma a la vez que la diseñaba. Se entiende bien con un ejemplo, la idea de poder puntuar los juegos que has añadido a una lista personal de favoritos no estaba concebida desde un inicio, pero al ver la necesidad de tener una entidad Usuario-Juego en la aplicación, simplemente había que añadir una valoración en forma de entero a dicha entidad y listo; el único problema a partir de ahí es la representación y captura de los datos, pero el modelo está listo para recibirlos y almacenarlos.

Así pues, podríamos decir que tenemos unas entidades base que he mencionado antes y unas entidades específicas, como son *Juego-Habilidad*, *Usuario-Juego* y *Usuario-Rol*. Las cuales detallan la relación entre sus dos entidades. Por ejemplo, si el día de mañana quisiéramos que la aplicación capturara la fecha y hora de cuando un usuario agrega un juego a favoritos, simplemente añadiríamos una variable de tipo fecha en *Usuario-Juego* y la instanciaríamos en el constructor.

Siguiendo el diagrama de entidades que se adjunta en el **Anexo I** "Entidades BBDD y Conexión con servidor externo" y como se aprecia en la siguiente tabla de índices, podemos comprobar cómo somos nosotros los encargados de cuidar las relaciones entre las entidades. Así pues, por ejemplo, dentro de nuestro controlador de editorial, a la hora de borrar una editorial, deberemos comprobar si no existen juegos con dicha editorial e impedir de esta manera su borrado. La flexibilidad que ofrece esta manera de trabajar exige prestar mucha atención a la lógica con la que se efectúan las operaciones.

Tabla de índices generados en MySQL Workbench.

Table	Name	Unique	Index...	Index Comment	Column
juego_habilidad	PRIMARY	Yes	BTREE		relacion_id
user	PRIMARY	Yes	BTREE		id
user_roles	PRIMARY	Yes	BTREE		roles_id
juego_habilidad	FKrd9nasufe18ly38x1hj9rwu0o	No	BTREE		juego_id
juego	FKor2fwpq5s8x9d2c5vns55h978	No	BTREE		editorial_id
editorial	PRIMARY	Yes	BTREE		editorial_id
user_juegos	FKpm6jk3pshdb841ayhd8gf4ph5	No	BTREE		user_id
user_juegos	FKfucsc2damufl903c30v778jbv	No	BTREE		juego_juego_id
user_juegos	PRIMARY	Yes	BTREE		id
role_web	PRIMARY	Yes	BTREE		id
juego	PRIMARY	Yes	BTREE		juego_id
juego	FKh6880lsdr16sj5q9hn9rjeowv	No	BTREE		categoria_id
categoria	PRIMARY	Yes	BTREE		categoria_id
habilidad	PRIMARY	Yes	BTREE		habilidad_id
juego_habilidad	FKivo8l5l9s5s0c2oagp5m5bc7a	No	BTREE		habilidad_id
user_roles	FK7mjx104dnyuojum4og2eyktde	No	BTREE		roles_id
user_roles	PRIMARY	Yes	BTREE		users_id

3. ASPECTOS FUNCIONALES Y DISEÑO DE LA APLICACIÓN

Aspectos funcionales

El principal objetivo de la aplicación es poder brindar y registrar información de una forma eficiente y sencilla. Para ello me planteé desde un inicio casos de uso que quería que mi aplicación pudiese cubrir y que pudiesen ser desarrollados en ciclos cortos.

Dentro de los casos de uso **básicos** tendríamos:

- Ingreso, registro y categorización de usuarios.
- Registro, edición, borrado y consulta de juegos, editoriales, categorías y habilidades específicas de cada juego.
- Consulta, edición y borrado de habilidades generales.

Los casos de uso más **específicos** contemplados en un principio son:

- Búsqueda básica por nombre aproximado y búsqueda avanzada con más criterios y filtros.
- Mostrar juegos de forma aleatoria en la vista principal, además de permitir al usuario refrescarlos.
- Lista de favoritos por usuario con la puntuación por juego correspondiente.
- Página con información sobre el proyecto y sus creadores.
- Avisto de cookies y página con las políticas de privacidad y uso de la web.
- Paginación del listado completo de juegos.
- Posibilidad de borrado de cuenta de usuario.

Las opciones de filtrado y búsqueda avanzada en conjunto a la vista principal en formato *kanban,* bajo mi punto de vista son las características más atractivas que el usuario encontrará en la web.

Para llevar a cabo estas funcionalidades, *Spring Boot* nos brinda el grueso de la configuración inicial de la aplicación, añadiendo los módulos que se han previsto utilizar. Esto sin embargo no condiciona que luego puedas añadir más módulos. En mi caso tuve que añadir manualmente el servicio *Spring Mail* de contacto, **funcionalidad** que no tenía prevista en un inicio.

Con el esqueleto de la aplicación construido, es decir, un **controlador**, las clases **modelo** de las principales entidades y unas **vistas** prototipo funcionales para observar los primeros resultados, empecé el diseño.

Diseño de la aplicación

He seguido una filosofía de diseño **OKR** (*Objective Key Result*), es decir, como si se tratase de una pirámide me planteé diseñar ciertas funciones básicas que representarían los cimientos del proyecto. Esta base la formarían operaciones de los registros de usuario y las operaciones *CRUD* implementadas por mis *DAO* que se materializan en los paquetes de **servicio** en el proyecto.

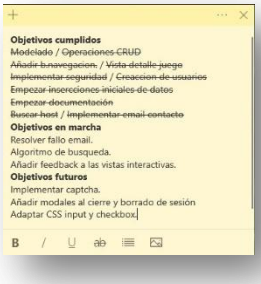
Con estas metas logradas, me planteé las siguientes funcionalidades: realizar las vistas de formulario y listas de las entidades, implementar una barra de navegación en la página principal y aplicar los niveles de seguridad a los distintos roles de usuario. En este punto del diseño se tomaron decisiones a nivel estético, como la disposición de los elementos del menú y el color insignia del portal o la forma de los logotipos.

Mientras iba desarrollando funcionalidades iba creciendo el proyecto y me surgían nuevas ideas, debido en parte a que no me encontraba grandes dificultades para avanzar. Así por ejemplo me decidí a implementar un "*captcha*" en el registro, un servicio de email para el contacto o todo el apartado legal del uso de la web.

La última parte del diseño ha sido la de perfilar detalles. Por poner un ejemplo, el botón en forma de corazón sobre la portada de los juegos para agregar a favoritos, está hecho a mano en *PSCS 6*, o el hecho de que cada vez que solicites la vista principal te aparezcan juegos de forma aleatoria son objetivos establecidos y ejecutados sobre la marcha.

En la organización de estos objetivos me he ayudado de las notas de *Windows* que ha sido suficiente para mi planificación. En un inicio se creó un canal de *slack* (herramienta que sirve para organizar el trabajo y la comunicación en equipos de desarrollo) para el proyecto, ya que éramos dos desarrolladores los que inicialmente trabajaríamos sobre el mismo. Quiero agradecer desde aquí a mi compañero Alex, ya que sin él, el inicio de este proyecto no habría sido el mismo.

Planificación por objetivos.



4. ASPECTOS DESTACABLES EN EL DESARROLLO

El patrón de construcción de los modelos es el mismo en toda la aplicación. Construimos un repositorio extendiendo ***JpaRepository***, en él implementamos las operaciones sobre la base de datos usando ***hql*** y las *palabras clave* soportadas por *Spring*, como por ejemplo, *findByVariableGreaterThan*.

Luego construimos nuestro **servicio** (@Service) en el cual instanciaremos nuestro **repositorio** (al igual que en el controlador usaremos @Autowired para esta operación) y desarrollaremos los métodos que interactúan con el mismo.

Finalmente creamos un **controlador** específico para cada entidad con su respectivo servicio instanciado, de tal manera que las operaciones ***GET y POST*** de dicho servicio quedan subyugadas a las solicitudes URL que recibe el servidor mediante un sufijo (HTTP REST).

Intentaré explicarlo más detalladamente con un ejemplo. Imaginemos que ya tenemos creada una lista de editoriales a las cuales accedemos en el controlador principal mediante la siguiente URL "localhost:8080/index/editoriales". Si analizamos esta petición al servidor, de forma interna, estamos accediendo a la raíz (es decir, el controlador principal) a través de '/index' y le estamos diciendo a este controlador además, que queremos más concretamente la función '/editoriales'.

Esto funciona así en la aplicación pero, ¿qué pasa con funciones específicas a las que solo deberían poder acceder los usuarios con un rol administrador, como por ejemplo, borrar, editar o crear nuevas editoriales? Aquí interviene el sistema de seguridad de *Spring*, el cual nos permite identificar a un usuario (con su rol incluido) y acotar las peticiones que puede realizar al servidor. De esta manera, si creamos un controlador específico para editorial en el que le añadimos como prefijo "/admin/" podremos incluir ahí todas aquellas operaciones que deseamos que solo un usuario con el rol establecido como administrador pueda realizar.

Así pues, con mi prototipo empecé a hacer pruebas en la configuración de la seguridad. Hay muchos detalles interesantes como el cifrado de la contraseña y la configuración del servicio *captcha*, pero me centraré en la configuración necesaria para el funcionamiento de los roles de usuario.

El epicentro de dicha configuración se encuentra en el paquete raíz de la aplicación. Es importante mencionar que el uso de los paquetes es esencial en *Spring*, todo parte de un paquete

raíz donde tenemos que tener nuestra clase lanzadera anotada con "*@SpringBootApplication*" y es aquí donde tenemos nuestra clase de configuración de seguridad, la cual debe extender a la clase "*WebSecurityConfigurerAdapter"*. El método "*configure"*, que se encuentra en dicha clase y donde mediante el objeto *"HttpSecurity"*, vamos a asegurar el acceso tanto a la petición *'/login*', *'/register'* como a *'/legal'* y por supuesto al archivo CSS asociado a las vistas para cualquiera que conecte con la aplicación.

Por otro lado, cualquier petición que contenga *'/admin/'*, solo será atendida si se realiza por un usuario que tenga el rol administrador, al igual que cualquier otro tipo de petición no será atendida si el usuario no ha conseguido ingresar correctamente (**ver Anexo II** "Configuración Seguridad").

De la construcción de las vistas quiero destacar, que si a día de hoy tuviese que decidir nuevamente entre *thymeleaf* o *jsp* seguramente me decantaría por el segundo ya que muchos de los ejemplos que he seguido para entender el funcionamiento de los módulos de *Spring* tenían como representación plantillas *jsp*, la cual es una tecnología que tiene mucho más recorrido porque lleva en el mercado muchos más años.

Thymeleaf al parecerse tanto a *HTML* se aprende relativamente rápido (los nombres de las etiquetas son los mismos con "*th:*" delante), además si nuestra aplicación está corriendo y solo modificamos las plantillas, no es necesario volver a desplegarla, esto último es tremendamente funcional (**ver Anexo III** "Ejemplo Thymeleaf").

He usado plantillas *Bootstrap 3* para la gran mayoría de componentes y elementos. Además algunos pequeños scripts en *JavaScript*, como para la valoración de favoritos en el que mediante una ventana modal le asignas una puntuación a alguno de tus juegos añadidos a dicha lista (**ver Anexo IV** "Ejemplo Bootstrap y JavaScript").

La barra de navegación de la página principal ("*index*") está desarrollada con *Bootstrap 4* que incorpora nuevas clases para dar estilos a los botones, cabeceras y tablas.

He implementado *google analytics*, este básicamente es un script que ha de colocarse el primero en la sección head de los archivos *HTML* que conforman las vistas de la aplicación. Nos otorgará mucha información acerca del uso y de los visitantes de la web (**ver Anexo V** "Portal Google Analytics").

5. INTERFAZ DE USUARIO

He seguido las tendencias actuales del diseño web para las vistas, se ha buscado la sencillez y un aire desenfadado para captar a un público no especializado.

Se ha intentado que tanto la información de navegación, el lenguaje de dicha información, los controles de usuario, la consistencia, así como el diseño de ayudas y páginas de error cumplan en la medida de lo posible con los principios de usabilidad de *Jakob Nielsen*.

En un principio me planteé un estilo "*responsive*" para poder acceder a la aplicación desde terminales móviles, pero el hecho de no centrarme en el despliegue hasta no tener muy desarrollada la aplicación y la consecuente imposibilidad de hacer pruebas en caliente, han hecho que me centrase totalmente en una aplicación web para escritorio. La experiencia es un grado, si el día de mañana tengo que desarrollar otra aplicación web voy a intentar desplegarla desde versiones tempranas, lo que me permitirá evaluar el progreso del diseño no solo desde mi equipo de forma local, sino también desde otros entornos y sistemas.

Se ha reciclado la barra de navegación para el uso tanto de administradores como de usuarios. En función del rol de dicho usuario aparecerán opciones de edición o de visionado. Además se han incluido *tooltips* en todos los enlaces no textuales de dicha barra para describir su función.

El resto de páginas *HTML* que componen el proyecto disponen de una barra de navegación más simple con un registro de dónde te encuentras y opciones para acceder tanto al contacto, como a la página principal nuevamente.

Reseñar que se ha utilizado un formato tipo "*ecommerce*" con las portadas de los juegos como reclamo para el usuario, un efecto "ficha de jugador" para describir las características de dichos juegos, ventanas modales para facilitar ciertas operaciones en el uso de la aplicación y la paginación en la lista de juegos para facilitar su visionado y uso.

Vista principal de la aplicación

6. DESPLIEGUE

Una de mis prioridades al plantearme el despliegue, ya que no es un proyecto al uso y no se va a monetizar, era que fuese a través de servicios gratuitos de hosting para aplicaciones web, también conocidos como *PaaS* (*Plataform as a Service*).

Decidido a encontrar este servicio me topé con *Heroku*, el cual está basado en contenedores (al igual que todas las ofertas de este tipo de servicios). Tras registrarme e instalar el cliente, intenté hacer mi primera subida sin éxito. El problema estaba en que faltaba "*MySQL*" en mi servidor, y ahí es donde descubrí que *Heroku* no servía para mis propósitos.

Para que te ofrezcan el servicio gratuito de *MySQL* y muchos otros, necesitaba obligatoriamente introducir los datos de una tarjeta de crédito. El sistema es sencillo, ellos no te cobran a menos que excedas una cuota de tráfico, esta es una práctica muy habitual en todos estos servicios. Como no quería monitorizar constantemente el tráfico y además he oído hablar de casos que tras un ataque se ha disparado el uso de los servicios de cobro, esta no era una opción para mí.

Así pues, encontré la plataforma Pivotal Web Services que operan con Cloud Foundry que me ofrece todos los servicios que buscaba de forma gratuita.

El proceso también es muy sencillo, te registras, instalas el cliente, activas el servicio *MySQL* y el contenedor ya está listo para recibir la imagen que crearemos mediante *Maven* y a la cual antes de crear le añadimos el archivo *manifest.yml* en la raíz del proyecto. Un detalle importante a destacar es que nuestra configuración de *Spring Data Jpa* tiene que estar en modo "*create*", ya que vamos a construir nuestra base de datos a partir de ella.

Desde el *PowerShell* o la consola de comandos una vez instalado el cliente ya podemos entrar a los servicios de *Pivotal Cloud Foundry*. Para ello se usa el comando "*cf login -a api.run.pivotal.io*", tras introducir nuestro usuario y contraseña nos moveremos a la carpeta donde tenemos la imagen creada (es decir, nuestra carpeta de proyecto) y con un simple "*push cf*" y el nombre asignado a la imagen, el cliente se pondrá a subir y levantar el servicio (**ver Anexo VI** "Despliegue Cloud Foundry").

Desde el portal web de *Pivotal Cloud Foundry* podemos observar nuestro servicio en ejecución, la cantidad de memoria que ha consumido y el tráfico recibido, además de consultar nuestro dominio asignado, en nuestro caso "https://playmesa.cfapps.io/" (**ver Anexo VII** "Portal Pivotal Cloud Foundry").

Ahora solo nos queda conectarnos a nuestro servidor *SQL* mediante el *Workbench* para poder administrarlo. Los datos de conexión los obtendremos a través del administrador web del servidor *SQL*, cuya dirección nos la proporciona el portal Web de *Pivotal* (**ver Anexo I** "Entidades BBDD y Conexión con servidor externo").

Para desplegar una nueva versión de la aplicación crearíamos la imagen con *Maven intall* y la subiríamos nuevamente con *cf push*. Hay que recordar que se borrarán los datos de nuestro servicio *SQL* ya que tenemos nuestra aplicación en "*create*". Así pues, antes de hacer una subida hacemos una copia de los datos mediante el *MySQL Workbench* (**ver Anexo VIII** "Backup MySQL Workbench").

7. COTROL DE VERSIONES Y SEGUIMIENTO

Para el control de versiones y mantenimiento del código he utilizado la plataforma *Github*. En un inicio se pensó en dos ramas de trabajo para cada desarrollador, pero una quedó en desuso ya que mi compañero no pudo continuar. Actualmente el código se encuentra almacenado en un repositorio privado perteneciente a una cuenta de proyecto y además, en mi cuenta pública personal (**ver Anexo IX** "Commits PlayMesa").

8. LÍNEAS DE TRABAJO FUTURO

Para continuar con el trabajo pondría esta pila de objetivos con este orden de sencillez y prioridad, siendo esta flexible al cambio y a las nuevas ideas que se puedan integrar.

Crear un rol "*SuperAdmin*". Las funciones que hago manualmente a través de *MySQL* Workbench, como pueden ser modificar el rol de un usuario o borrarlo, se pueden llevar a cabo a través de la propia aplicación. Esta es una tarea sencilla que haría más cómoda la administración de la aplicación y no la haría dependiente del desarrollador.

Crear un acceso en modo "demo". Este sería más un vídeo demostrativo que un acceso real al servicio, y permitiría ver las posibilidades del mismo sin necesidad de registro o demostración manual.

Servicio de recuperación de contraseña automático además de la confirmación de registro. Esto que a simple vista parece un sencillo paso requería transformar nuestro módulo de seguridad en algo más real y práctico, habría que implementar un *OAuth2* o similar.

Empezar a buscar colaboración exterior y contacto con centros reales interesados. En este punto podríamos decir que la aplicación ya tendría una funcionabilidad plena y habría que plantearse ampliar el servicio *host*, así como automatizar un salvado de datos semanal o diario.

9. CONCLUSIONES

Este proyecto me ha ayudado a poner en práctica muchos de los conocimientos adquiridos durante el curso y a ir un poco más allá. El uso de un "*framework*" y el despliegue de una aplicación en un servicio *Cloud* me parecen campos muy interesantes e imprescindibles en el abanico de conocimientos de un desarrollador. En mi caso es un servicio *Web*, pero este mismo proceso sería equivalente *en micro-servicios* (actualmente de lo más implementado y solicitado en el sector) o para trabajar en arquitecturas *Serverless*.

Hoy en día ya nadie programa sin "*frameworks*". Esta frase la he escuchado varias veces en clase y quiero invitar a una reflexión a través de ella. A la vez que estas herramientas nos hacen la vida más cómoda y fácil, tienen peligros ocultos a los cuales no hay que perder de vista. Nos alejan del "*core*", nos obligan a adaptarnos tanto a las posibilidades del mismo como a su devenir y por lo general se cumple la norma de que a mayor abstracción menor rendimiento.

Tecnologías como *Electron* que permiten el desarrollo de aplicaciones gráficas de escritorio con entornos puramente web (sus aplicaciones corren bajo un navegador, integrado sí, pero un navegador al fin y al cabo), no nos permiten el control sobre sus threads, no permiten tampoco compartir memoria entre procesos y la división de estos la hace como él considera más oportuna. ¿Hasta qué punto, como programadores, estamos dispuestos a sacrificar control y capacidad creativa a cambio de comodidad y facilidades de integración?

La situación actual en el desarrollo web creo que necesita renovarse y que cambiará en los próximos años, después de todo nadie puede ser un experto en *angular*, *react*, *laravel*, *vue*, *ruby on rails* y *symphony* a la vez.

Hace unos años si querías convertir un archivo *mp3* a formato *wav* tenías que bajarte un software específico, hoy recurres a un servicio *web*. Después de todo, no hay que olvidar que estamos en un momento en el que los *SaS* triunfan frente a las distribuciones clásicas de aplicaciones, que tanto el desarrollo web como los navegadores avanzan a ritmos desmesurados y que quizás, esa clásica línea divisoria entre el *backend* y *frontend* cada día está más difuminada.

Ahora mismo podemos atisbar las puertas de un futuro prometedor en el cual la necesidad del *hardware* a nivel usuario se puede quedar en entredicho. Gracias entre otros muchos avances, a la red *5G* (10,000 *Megabits por segundo* de tasa de transferencia) que viene para cambiar el paradigma de las **comunicaciones** y el **procesamiento de datos**.

10. BIBLIOGRAFÍA / ENLACES DE INTERÉS

Para la realización de esta aplicación se han consultado la siguiente documentación y webs:

Documentación Spring Oficial

Foro StackOverFlow

W3C Bootstrap

Plantillas HTML y CSS

Thymeleaf

Cursos de formación OpenWebinars

Baeldung Spring

Base de datos juegos de mesa BGG

Tutorials Point Spring Security

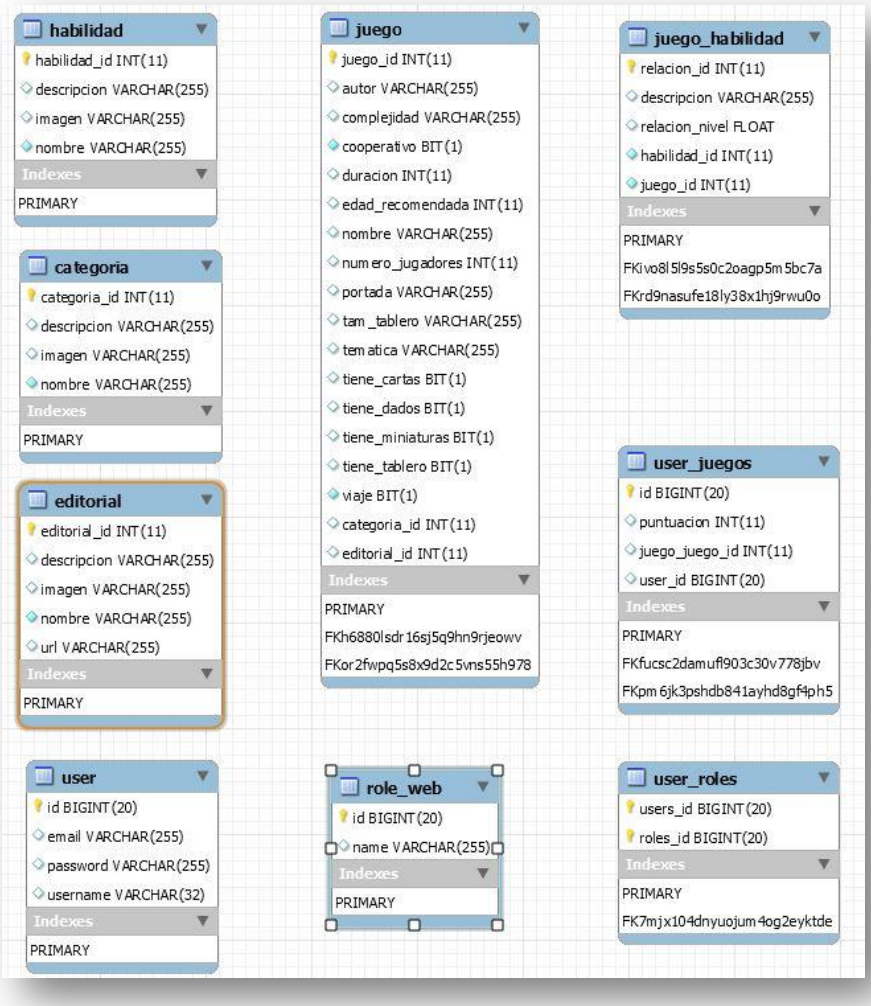
Tienda juegos de mesa

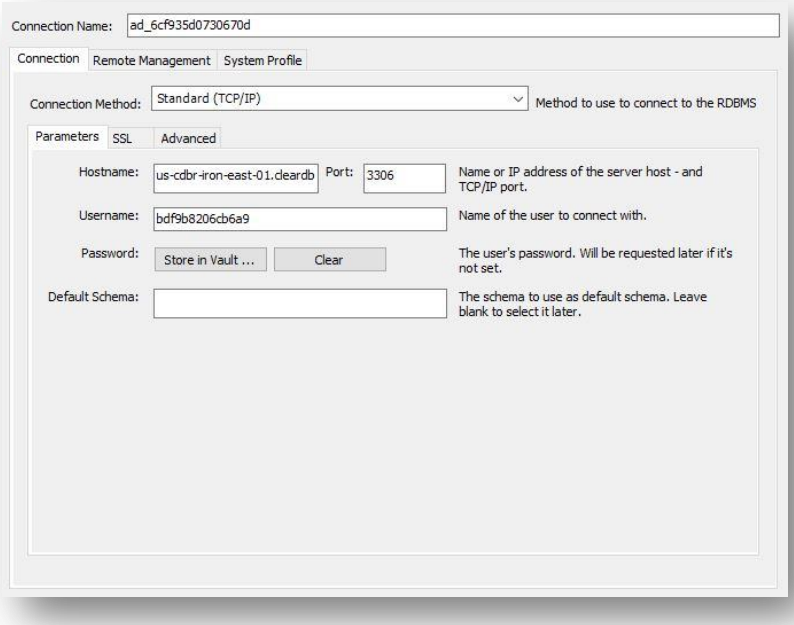
Pivotal Cloud Foundry Tutorial

Libro Mastering Spring Ranga Rao Karanam

11. ANEXOS

ANEXO I Entidades BBDD y Conexión con servidor externo.





ANEXO II Configuración Seguridad

```java
package com.playmesa;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Qualifier("userDetailsServiceImpl")
    @Autowired
    private UserDetailsService userDetailsService;

    @Autowired
    PasswordEncoder passwordEncoder;

    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(bCryptPasswordEncoder());
    }

    @Override
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

     @Bean
     public PasswordEncoder passwordEncoder() {
          return new BCryptPasswordEncoder();
      }
        @Override
        protected void configure(HttpSecurity http) throws Exception {

            http
            .authorizeRequests()
                .antMatchers("/resources/**", "/registration","/legal", "/css/style.css", "/registration/submit").permitAll()
                .antMatchers("/admin/**").hasRole("ADMIN")
                .anyRequest().authenticated()
                .and()
            .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
            .logout()
                .permitAll()
                .and()
                .csrf().disable();

        }
}
```

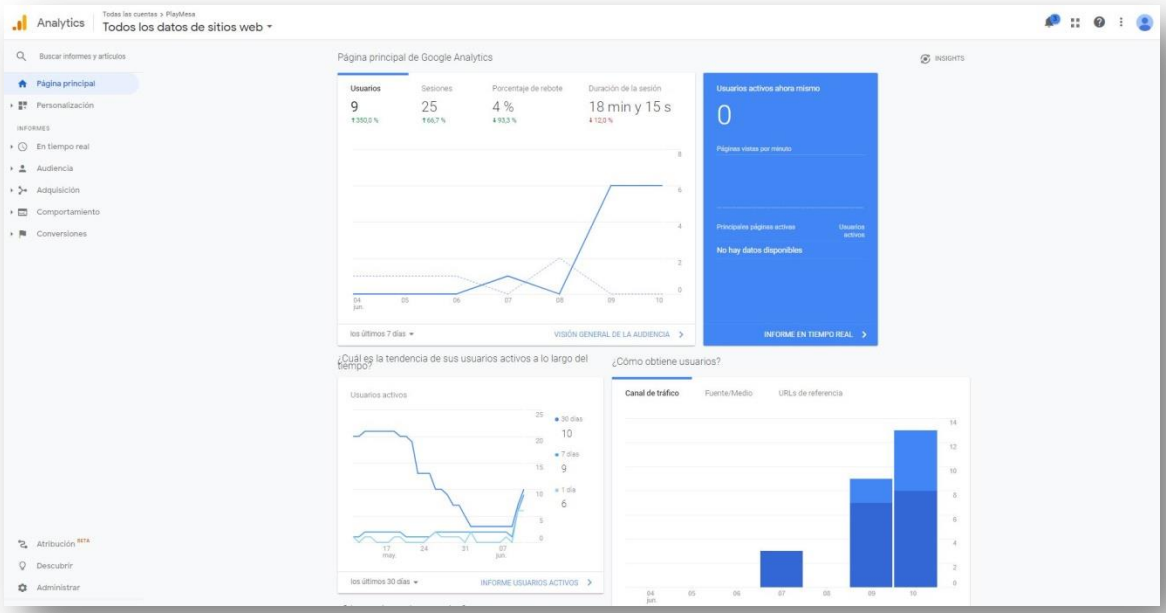
ANEXO III Ejemplo Thymeleaf

```
<!DOCTYPE html>
<html lang="es" xmlns:th="http://www.thymeleaf.org">
<head>
<!-- Global site tag (gtag.js) - Google Analytics -->
<script async src="https://www.googletagmanager.com/gtag/js?id=UA-164001435-1"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());

  gtag('config', 'UA-164001435-1');
</script>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap -->
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
    th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" rel="stylesheet">
<link href="../../static/css/style.css" th:href="@{/css/style.css}"
    rel="stylesheet" />

</head>
<body>
    <header>
        <nav class="navbar navbar-default">
      <div class="container-fluid">
        <div class="navbar-header">
          <a class="navbar-brand" href="#" th:href="@{/index}">PlayMesa</a>
        </div>
        <ul class="nav navbar-nav navbar-right">
          <li class="active"><a href="#" th:href="@{/admin/editorial/}">Lista Editoriales</a></li>
          <li><a href="#" th:href="@{/contacto/}">Contacto</a></li>
          <li><a href="#" th:href="@{/logout}">Cerrar Sesión</a></li>
        </ul>
      </div>
    </nav>
    </header>
    <div class="container">
        <div class="row">
            <div class="col-md-offset-2 col-md-8">
                <table class="table">
                    <thead>
                        <tr>
                            <th>Imagen</th>
                            <th>Nombre</th>
                            <th>Descripción</th>
                            <th>Web</th>
                            <th>Total Juegos</th>
                        </tr>
                    </thead>
                    <tbody>
                        <tr th:each="editorial : ${editoriales}">
                            <td><a th:href="@{/index(idEditorial=${editorial.editorialId})}"><img
                                th:src="${not (#strings.isEmpty(editorial.imagen))} ? ${editorial.imagen} : 'http://placehold.it/48x48'"
                                class="img-responsive icono-categoria" alt="imagen" /></a></td>
                            <td><a type="button" th:href="@{/index(idEditorial=${editorial.editorialId})}" th:text="${editorial.nombre}"></a></td>
                            <td th:text="${editorial.descripcion}"></td>
                            <td><a target="_blank" th:text="${editorial.url}" th:href="${editorial.url}"></a></td>
                            <td th:text="${editorial.juegoList.size()}"></td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>


    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"
        th:src="@{/webjars/jquery/jquery.min.js}"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
        th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"></script>

    <script src="../../static/js/app.js" th:src="@{/js/app.js}"></script>


</body>
</html>
```

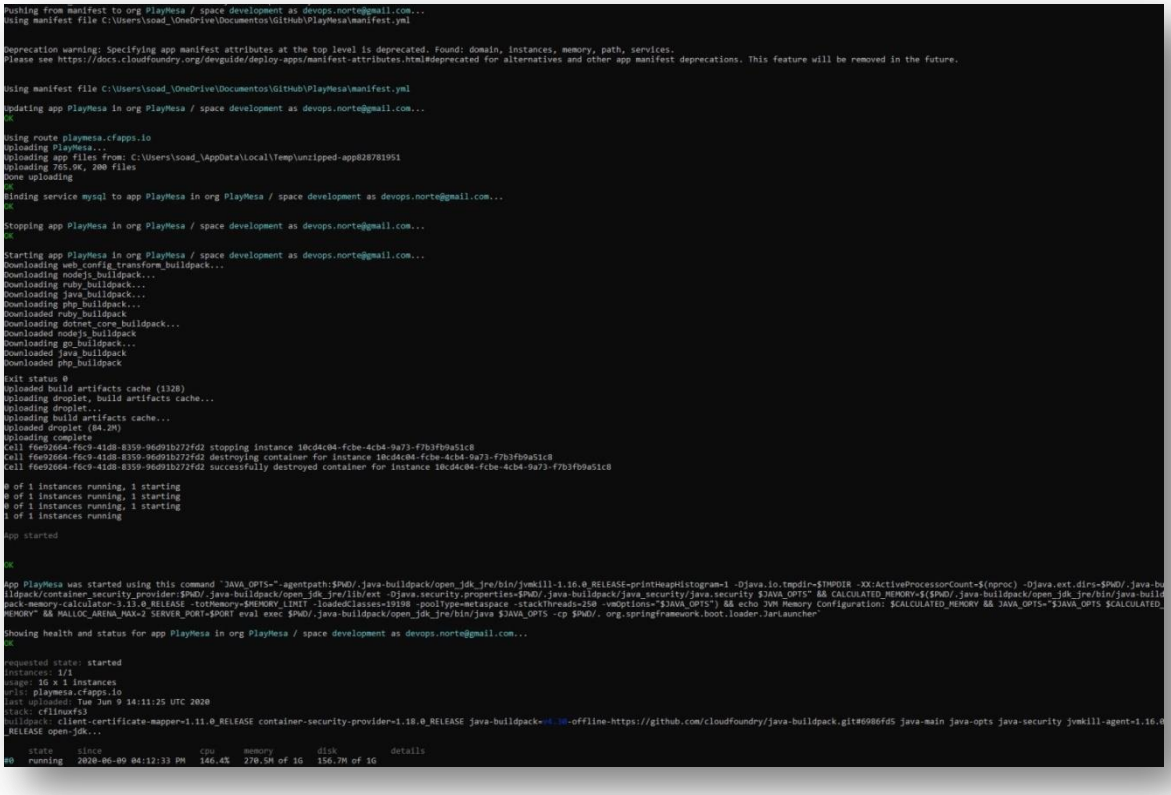
ANEXO IV Ejemplo Bootstrap y JavaScript

```
<div class="row">
    <div class="col-sm-2" th:each="juego : ${juegos}">
        <div class="col-item">
            <div class="photo">
                <img src="http://placehold.it/350x260"
                    th:src="${#strings.isEmpty(juego.portada)} ? 'http://placehold.it/350x260' : ${juego.portada}"
                    class="img-rounded" alt="Cinque Terre" width="304" height="236">
            </div>
            <div class="info">
                <div class="row">
                    <div class="juego col-md-12">
                        <h5 class="titjue">
                            <a th:href="@{/juego/{juegoId}(juegoId=${juego.juegoId})}"
                                th:text="${#strings.abbreviate(juego.nombre,65)}">Nombre</a>
                        </h5>
                        <a
                            th:href="@{/favorito(juegoId=${juego.juegoId},username=${#authentication.name})}"
                            class="heartbox"> <img border="0" alt="agregar favorito"
                            src="https://i.ibb.co/wp0ZjF3/like.png"
                            onmouseover="this.src='https://i.ibb.co/KmNMZvc/likehover.png'"
                            onmousedown="this.src='https://i.ibb.co/BGxg7nS/likeactive.png'"
                            onmouseout="this.src='https://i.ibb.co/wp0ZjF3/like.png'">
                        </a>
                        <div style="text-align: center">
                            <a class="linksps" th:text="${juego.editorial.nombre}"
                                th:href="@{/index(idEditorial=${juego.editorial.editorialId})}"></a>
                            - <a class="linksps" th:text="${juego.categoria.nombre}"
                                th:href="@{/index(idCategoria=${juego.categoria.categoriaId})}"></a>
                        </div>
                    </div>
                    <div class="row"></div>
                </div>
            </div>
        </div>
    </div>
</div>

$(document).ready(function() {
    $('#delete-modal').on('show.bs.modal', function(event) {
        var button = $(event.relatedTarget);
        var data = button.data('id');
        var modal = $(this);
        var a = modal.find('.modal-body a')[0];
        a.href = a.href + data;
    });
});

$(document).ready(function() {
    $('#value-modal').on('show.bs.modal', function(event) {
        var button = $(event.relatedTarget);
        var data = button.data('id');
        var modal = $(this);
        var a = modal.find('.modal-body a')[0];
        a.href = a.href + data;
    });
});


$(document).ready(function() {
    $('#punt').click(function() {
        var a = document.getElementById("punt");
        var input = document.getElementById("ranking");
        a.href = a.href + '/' + input.value + '/';
    });
});
```

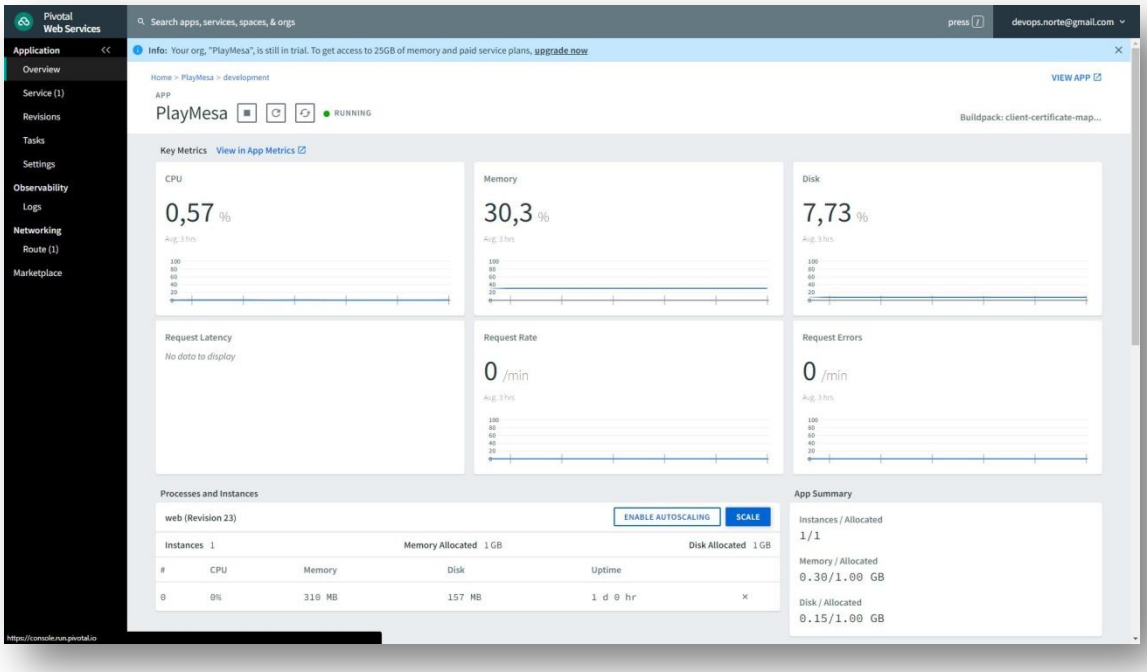
ANEXO V Portal Google Analytics



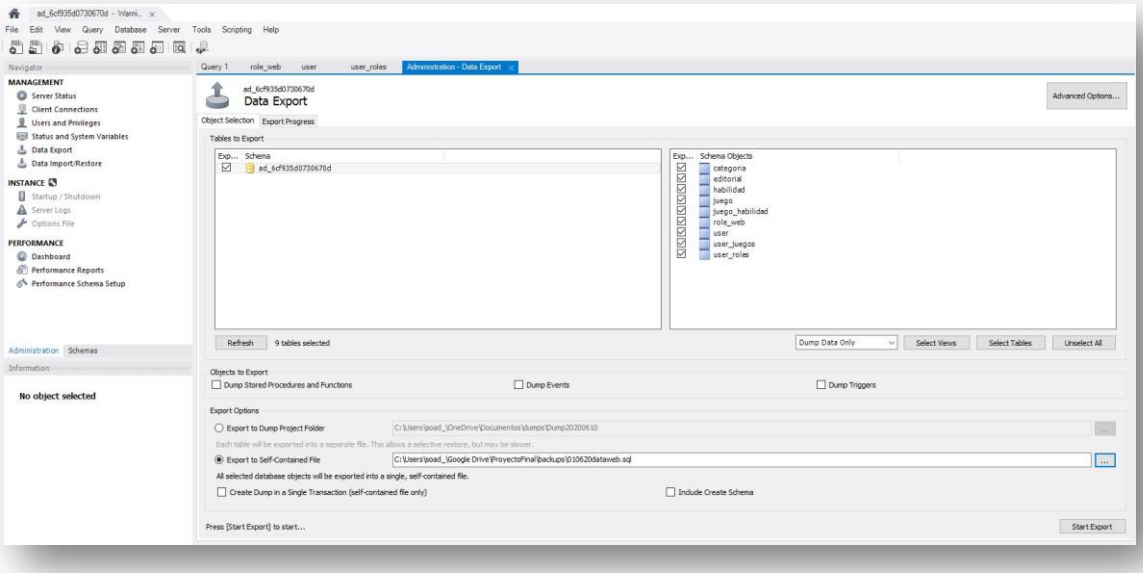
ANEXO VI Despliegue Cloud Foundry



ANEXO VII Portal Pivotal Cloud Foundry



ANEXO VIII Backup MySQL Workbench



ANEXO IX Commits PlayMesa

```
$ git log
commit c0c98e520250c8a7b0b9ccfbfe6297e9153f7aee
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Mon Jun 1 21:44:39 2020 +0200

    Se añade el borrado de cuenta

    Se añade el borrado de cuenta, se actualizan modales y condiciones de busqueda

commit 7463653cdf975a3cd48d5b048b7cb1b3611347b7
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Wed May 20 10:22:10 2020 +0200

    commit 20may

commit 61b3db19fe699dad265b56844fd15f04de9779b5
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Fri May 1 15:56:23 2020 +0200

    v0.2 captcha, search and favorit

    Se implementa la busqueda avanzada, la comprobación del captcha y la puntuación de favoritos

commit 6053f1e7bf4d3b82f3cd95037de6ff5591ba983a
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Thu Apr 23 13:08:07 2020 +0200

    PlayMesa v0.1

    Busqueda, favoritos, registro con captcha version de despliegue.

commit 21109e3d6cb5c70e2ba35503e65ea78fad94212b
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Thu Apr 16 21:16:31 2020 +0200

    commit de despliegue

commit 9dcac3264c7157d5a7c5e6824b904acf354ab190
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Fri Apr 10 12:48:22 2020 +0200

    nuevas features e información

    Nuevas funcionalidades

commit 9daa315a08ac2d3d8b0bdb9b8bbc44ad769b4b79
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Mon Apr 6 22:56:47 2020 +0200

    Springsecurity configurado, formularios y vistas añadidas

commit 0a443ef3dba3437475ef6add74a65f573bd9674b
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Fri Mar 20 12:27:32 2020 +0100

    Segundo commit, estructura terminada

commit 78f823c8d4fdb1ee7919db11ff8b5716e5b8a1d0
Author: PJ <pj.rivera.perez@gmail.com>
Date:   Wed Mar 18 13:53:06 2020 +0100

    Primer commit con estructura del proyecto
```

```
* c0c98e5 - (10 days ago) Se añade el borrado de cuenta - PJ (HEAD -> master, origin/master)
* 7463653 - (3 weeks ago) commit 20may - PJ
* 61b3db1 - (6 weeks ago) v0.2 captcha, search and favorit - PJ
* 6053f1e - (7 weeks ago) PlayMesa v0.1 - PJ
* 21109e3 - (8 weeks ago) commit de despliegue - PJ (heroku/master)
* 9dcac32 - (9 weeks ago) nuevas features e información - PJ
* 9daa315 - (9 weeks ago) Springsecurity configurado, formularios y vistas añadidas - PJ
* 0a443ef - (3 months ago) Segundo commit, estructura terminada - PJ
| * a6c884c - (3 months ago) Primer commit de prueba - playmesa (origin/playmesa_alex)
|/
* 78f823c - (3 months ago) Primer commit con estructura del proyecto - PJ
```